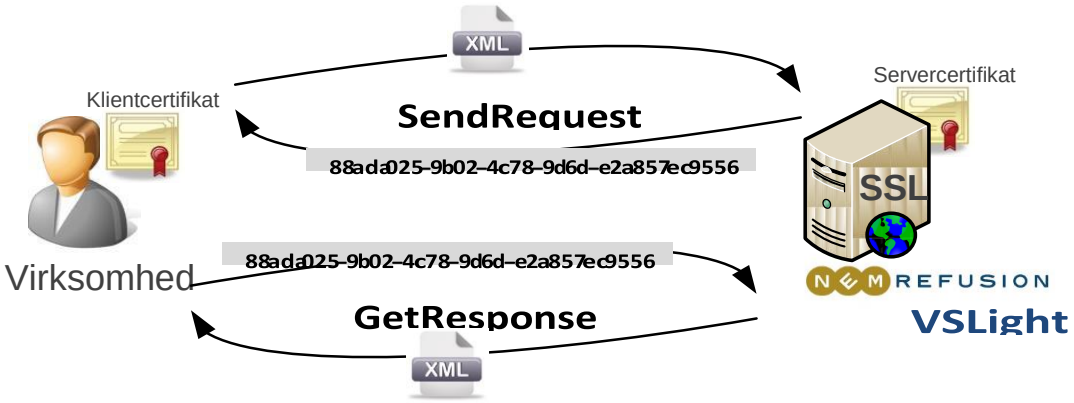
Virksomhedsservice via webservice

Dette dokument beskriver hvordan der forbindes til og udveksles beskeder med NemRefusions virksomhedsservice via Web servicen ***VSLight***.

Ud over de tekniske detaljer beskrives også et konkret kode eksempel på hvordan integrationen kan udføres med .NET Framework 3.5 og Visual Studio 2010.



INDHOLD

1 Forudsætninger

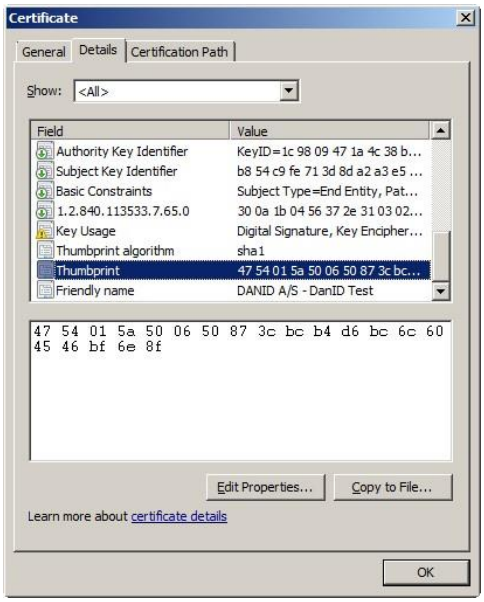
Dette dokument beskriver kun hvordan korrekt kommunikation etableres med VSLight på et teknisk plan. Detaljer omkring XML beskedformater og indberetningstyper skal indhentes via snitfladebeskrivelse og vejledninger på *www.nemrefusion.dk*.

1.1 Gyldig og signeret XML

VSLight erstatter kun selve MQ kommunikationen, så det er nødvendigt, at brugeren kan danne en gyldig NemRefusion virksomhedsservice XML besked (*NemRefusionIndberetningSamling*, *NemRefusionVirksomhedSoegningStruktur*, *HaendelseSoegningStruktur* eller *SagsbehandlingSoegningStruktur*) og korrekt påføre en XML signatur (XMLDsig).

1.2 Registrering af klientcertifikat

Ud over at klientcertifikatet der benyttes er gyldigt, skal certifikatets *thumbprint/miniatureudskrift* også være registreret hos VSLight servicen. Dette gøres ved først at finde denne værdi for det certifikat der ønskes benyttet under egenskaber for certifikatet i certifikatlageret, og herefter fremsende denne til NemRefusion.



2 Ressourcer

Placering af WSDL for VSLight servicen:

https://vslight.nemrefusion.dk/VSLight.svc?wsdl

Vær opmærksom på at denne inklusiv de underliggende referencer også påkræver klientcertifikat ved tilgang. Dette kan have den konsekvens at værktøjer til automatisk generering af kode kan have problemer med at køre. Hvis dette er tilfældet kan fremgangsmåden beskrevet i integrationseksemplet benyttes.

3 Beskrivelse

VSLight er en sikker (https + påkrævet "registreret" klientcertifikat) webservice (WCF) baseret på .NET Framework 4.6.2 som udstiller to metoder:

- string **SendRequest**(XElement XML); *"læg på kø"*
- XElement **GetResponse**(string RequestUUID); *"hent fra kø"*

Interfacet er typeløst for den transporterede XML så der kan ligesom på en kø kommunikeres enhver form for gyldig XML.

3.1 Metoden SendRequest

Metoden tager som input den NemRefusion XML besked der ønskes indsendt, og output er et UUID som skal benyttes ved senere afhentning af svar XML beskeden fra NemRefusion med metoden ***GetResponse***.

Inputtypen bliver ved integration fra .NET til et objekt af typen *XElement* (*System.Xml.Linq*) som kan indeholde en generisk XML struktur og som er nem at arbejde med. Hvis der f.eks. benyttes et værktøj som *SoapUI* vil XML input være af typen *<xsd:any>*. Output er en tekststreng som altid vil indeholde en gyldig UUID (.NET Guid i en *string*).

3.2 Metoden GetResponse

Efter et succesfuldt kald til ***SendRequest*** vil der, ideelt set, altid kunne hentes en svar XML besked ud igen, når denne er færdigbehandlet i NemRefusion. Hvornår svarbeskeden er klar vil variere alt afhængig af størrelsen på input og NemRefusions tilstand. Derfor kan det være nødvendigt løbende at forespørge på den sammen besked, hvis ikke svaret blev fundet ved første forespørgsel.

Input og output til metoden ***GetResponse*** fungerer bare omvendt af ***SendRequest***. Input er den UUID som blev modtaget ved kaldet til ***SendRequest*** og output er en XML besked med svaret fra NemRefusion.

Hvis metoden returnerer ***null*** så har servicen ikke kunne finde en XML besked med svaret fra BizTalk. Dette kan betyde, at forespørgslen ikke er behandlet færdig endnu (eller er stallet), og der må prøves igen lidt senere. Alternativt kan denne returværdi betyde, at det UUID der medsendes er forkert og altså aldrig har været returneret fra et kald til ***SendRequest***, eller det kan betyde at kaldet til ***GetResponse*** er foretaget for sent, og filen med svaret er blevet ryddet op (slettet eller arkiveret efter 14 dage).

3.3 Fejl

Gyldigt input til metoden *GetResponse* er en UUID, men typen er kun begrænset til en tekststreng og kan derfor være ugyldig. I da fald modtager servicekalderen fejlbeskeden (System.ServiceModel.FaultException)

```
RequestUUID does not contain a valid UUID. (Example: 98f44b5b-5a45-4ad9-b13e-4b9366e5c01f)
```

Hvis servicen har problemer med at læse eller skrive i de to delte mapper, som bruges til kommunikationen med BizTalk modtages fejlbeskederne:

```
Request XML could not be delivered to an underlying system. Please try again later.
```

```
Response XML could not be acquired from an underlying system. Please try again later.
```

4 Sikkerhed

Kommunikation med servicen er SSL krypteret og kan altså kun foregå over **HTTPS**. Endvidere skal klienten ved ethvert kald autentificere sig med et gyldigt certifikat på transportniveau. Certifikatet skal kunne fungere som klientcertifikat og samtidig må det ikke være udløbet eller spærret og det skal have en udsteder som er betroet af VSLight webserveren (f.eks. "TDC OCES CA").

Som beskrevet under forudsætninger for brug af servicen, ligger der en yderligere sikkerhed i, at klientcertifikater også skal være registreret lokalt hos VSLight.

5 Fejlbeskeder

Ved brug af servicen kan der forekomme en række fejlbeskeder som er beskrevet i følgende tabel.

Fejlkode	Fejltekst	.NET Exception type	Årsag/handling
101	RequestUUID does not contain a valid UUID. (Example: 98f44b5b-5a45-4ad9b13e-4b9366e5c01f)	System. ServiceModel. FaultException	Input til metoden GetResponse skal være et gyldigt UUID.
102	User with client certificate thumbprint #THUMBPRINT# is not a registered subscriber.	System. ServiceModel. FaultException	Certifikatets thumbprint/miniatureudskrift er ikke fremsendt og registreret hos VSLigth/NemRefusion.
103	Request XML could not be delivered to an underlying system. Please try again later.	System. ServiceModel. FaultException	Ved kald til *SendRequest* kunne den indsendte XML ikke leveres videre til NemRefusion systemet. Dette kan skyldes midlertidig nedetid og kaldet bør forsøges gentaget på et senere tidspunkt.

104	Response XML could not be acquired from an underlying system. Please try again later.	System. ServiceModel. FaultException	Ved kald til *GetResponse* kunne den ønskede XML ikke hentes fra NemRefusion systemet. Dette kan skyldes midlertidig nedetid og kaldet bør forsøges gentaget på et senere tidspunkt.
InternalServiceFault	The server was unable to process the request due to an internal error. For more information about the error, either turn on IncludeExceptionDetailInFaults (either from ServiceBehaviorAttribute or from the <serviceDebug> configuration behavior) on the server in order to send the exception information back to the client, or turn on tracing as per the Microsoft .NET Framework 3.0 SDK documentation and inspect the server trace logs.	System. ServiceModel. FaultException	Der er sket en uhåndteret fejl internt i VSLight servicen. Prøv igen senere og kontakt evt. NemRefusion hvis fejlen ikke forsvinder.
N/A	The HTTPS request was forbidden with client authentication scheme 'Anonymous'.	System. ServiceModel. Security. MessageSecurityException	Ugyldigt eller manglende certifikat. 1. Der er slet ikke påhæftet et klientcertifikat på servicekaldet. 2. Certifikatet er udløbet, spærret eller dets udsteder er ikke betroet af VSLight webserveren. 3. Certifikatets thumbprint/miniatureudskrift er ikke fremsendt og registreret hos VSLigth/NemRefusion.

6 Visual Studio 2010 eksempel

I det følgende beskrives et udførligt eksempel på hvordan der kan kommunikeres med VSLight fra en Visual Studio 2010 .NET Framework 3.5 konsolapplikation. Eksemplet beskriver hvordan man gør hvis man har en gyldig signeret NemRefusion virksomhedsservice XML besked liggende i en folder, og gerne vil have denne behandlet i NemRefusion og gemme svaret i en anden folder.

6.1 Lokal WSDL

Da Visual Studios "Add Service Reference…" funktionalitet og værktøjet *svcutil.exe* kan have problemer med ressourcer der kræver klientcertifikat for tilgang, kan det være nødvendigt med følgende indledende manøvre hvor alle WSDL ressourcer lægges i en lokal mappe.

Brug Internet Explorer til at downloade følgende filer en efter en og gem dem i samme mappe med det foreslåede filnavn fra tabellen herunder (det kan være nødvendigt at trykke på Alt-tasten for at få fil menuen og "Gem som" frem). Det gyldige registrerede klientcertifikat skal være tilgængeligt på maskinen og vælges når Internet Explorer spørger efter det.

Adresse	Gem som
https://vslight.nemrefusion.dk/VSLight.svc?wsdl	…/VSLight.wsdl
https://vslight.nemrefusion.dk/VSLight.svc?wsdl=wsdl0	…/wsdl0.wsdl
https://vslight.nemrefusion.dk/VSLight.svc?xsd=xsd0	…/xsd0.xsd
https://vslight.nemrefusion.dk/VSLight.svc?xsd=xsd1	…/xsd1.xsd

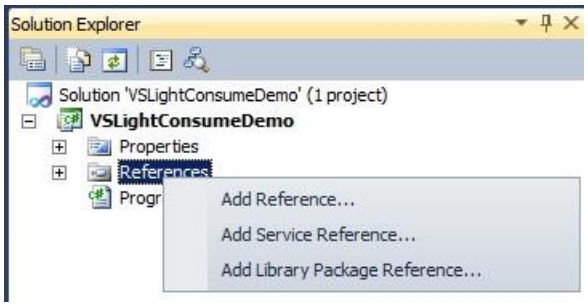
Når dette er gjort, skal referencerne inde i filerne erstattes med stier til de nye lokale placeringer. Derfor skal der redigeres i filerne som beskrevet i tabellen herunder.

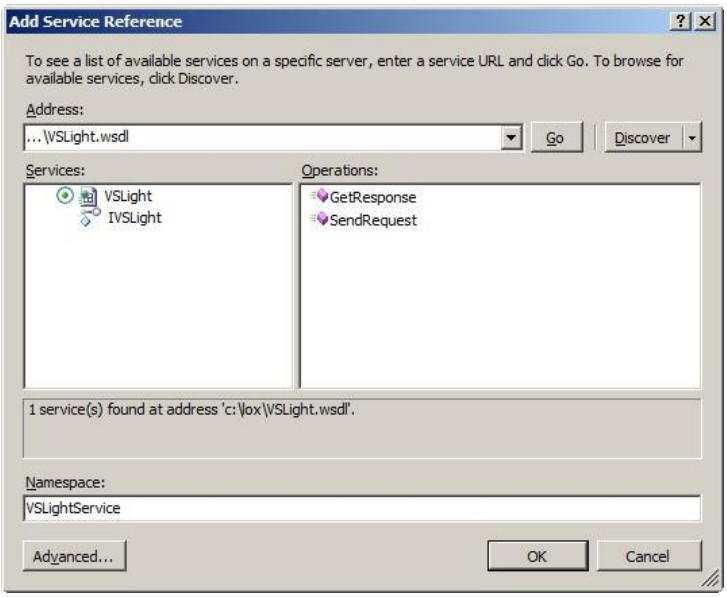
Fil	Erstat dette	Med dette
…/wsdl0.wsdl	"https://vslight.nemrefusion.dk/VSLight.svc?xsd=xsd0"	"xsd0.xsd"
…/wsdl0.wsdl	"https://vslight.nemrefusion.dk/VSLight.svc?xsd=xsd1"	"xsd1.xsd"
…/VSLight.wsdl	"https://vslight.nemrefusion.dk/VSLight.svc?wsdl=wsdl0"	"wsdl0.wsdl"

Herefter skulle filen "VSLight.wsdl" være til at konsumere.

6.2 Tilføj servicereference

Start Visual Studio og dan et nyt projekt af typen *Console Application* på minimum version 3.5 af .NET Framework. Vælg "Add Service Reference…", angiv placeringen af den lokale (eller eksterne) fil *VSLight.wsdl* og angiv f.eks. namespace som "VSLightService".





6.3 Angiv klientcertifikat

Det meste af den **information** der er nødvendig for at kommunikere med servicen er nu automatisk placeret i app.config filen (web.config ved webløsninger), dog mangler der en reference til et klientcertifikat.

```
...
<client>
  <endpoint address="https://vslight.nemrefusion.dk/VSLight.svc"
      binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IVSLight"
      contract="VSLightService.IVSLight" name="BasicHttpBinding_IVSLight" behaviorConfiguration="ClientCertificateBehavior" />
</client>
<behaviors>
  <endpointBehaviors>
    <behavior name="ClientCertificateBehavior">
      <clientCredentials>
        <clientCertificate storeLocation="LocalMachine" storeName="My" x509FindType="FindByThumbprint" findValue="..." />
```

```
      </clientCredentials>
    </behavior>
  </endpointBehaviors>
</behaviors> ...
```

En ny sektion med navnet *behaviors* skal indsættes som herover og værdien for *findValue* skal angives til certifikatets thumbprint/miniatureudskrift (f.eks findValue="47 54 01 5a 50 06 50 87 3c bc b4 d6 bc 6c 60 45 46 bf 6e 8f"). En reference til denne nye *endpointBehaviour* skal lægges ind som ny attribut i *endpoint* elementet i den ovenstående *client* sektion (behaviorConfiguration="ClientCertificateBehavior").

Denne konfiguration antager at certifikatet ligger i certifikatlageret for den lokale computerkonto i mappen "Personligt/Personal", men værdierne kan nemt justeres til at finde certifikater i andre lagre. Endvidere kan tilknytning af klientcertifikat også foretages direkte via kode.

6.4 Forøg grænsen for modtaget datamængde

Der er en indbygget grænse for hvor meget data der kan sendes til VSLight servicen på 10MB, men denne begrænsning foregår på server siden. Ved kald til *GetResponse* kan svaret indeholde store mængder data til klienten og en begrænsning på hvad der accepteres er per default sat i konfigurationsfilen. Derfor bør der justeres i *binding* sektionen sådan at værdierne for *maxBufferSize* og *maxReceivedMessageSize* er betydeligt større. Et forslag er som de markerede værdier herunder.

```
...
<bindings>
  <basicHttpBinding>
    <binding name="BasicHttpBinding_IVSLight" closeTimeout="00:01:00"
openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
        allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
maxBufferSize="65536000" maxBufferPoolSize="524288" maxReceivedMessageSize="65536000"
messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"         useDefaultWebProxy="true">
      <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
maxBytesPerRead="4096" maxNameTableCharCount="16384" />
      <security mode="Transport">
        <transport clientCredentialType="Certificate" proxyCredentialType="None"
realm="" />
        <message clientCredentialType="UserName" algorithmSuite="Default" />
      </security>
    </binding>
  </basicHttpBinding>
</bindings> ...
```

Det kan heller ikke udelukkes at det bliver nødvendigt at justere på timeoutværdier i *binding* sektionen.

6.5 Kode

Med en reference til det nye VSLight namespace (`using VSLightConsumeDemo.VSLightService;`) kan koden i de næste to afsnit bruges til at demonstrere de vigtigste aspekter af VSLight servicens to metoder med henblik på input og fejlhåndtering. Koncepterne demonstreret her skulle meget gerne hurtigt kunne omsættes til praktisk brug af VSLight.

6.5.1 Metoden SendXmlFilTilVSLight

```
/// <summary>
/// En signeret virksomhedservice XML besked i en fil sendes til NemRefusion.
/// </summary>
/// <param name="stiTilXmlFil">Komplet sti til XML fil</param>
/// <returns>Et UUID som skal benytte ved senere afhentening af svar
XML</returns> public static string SendXmlFilTilVSLight(string stiTilXmlFil)
{

   // XML læses fra disk til et XElement objekt. Her er det meget vigtigt at vælge
PreserveWhitespace!!!     var inputXml = XElement.Load(stiTilXmlFil,
LoadOptions.PreserveWhitespace);

   // opret VSLight klient
   var vsLightKlient = new VSLightClient();

try    {
      // returner svaret fra kaldet til SendRequest
return vsLightKlient.SendRequest(inputXml);
   }
   catch (MessageSecurityException mse) // sikkerhedsfejl
   {
      Console.WriteLine("Der er noget galt med klientcertifikatet: " + mse.Message);
   }
   catch (FaultException fe) // soap fejl med koder
   {
      switch (fe.Code.Name) // fejl skal håndteres ifølge fejlkoden
      {           case
"102":
            Console.WriteLine("Det benyttede certifikat er ikke registreret hos VSLight: " + fe.Message);
            // Håndter!
            break;
         case "103":
            Console.WriteLine("VSLight kunne ikke levere filen videre til NemRefusion, vi prøver igen senere: " + fe.Message);
            // Håndter!
            break;
case "InternalServiceFault":
            Console.WriteLine("VSLight må være nede, vi prøver igen senere: " + fe.Message);
            // Håndter!
            break;
default:
            Console.WriteLine("Uventet svarkode:" + fe.Message);
break;
      }   }
   catch (Exception e)
   {
      Console.WriteLine("Uventet fejl:" + e.Message);
   }
   finally // VSLight klienten skal altid lukkes
   {
      vsLightKlient.Close();
   }
   return null; }
```

6.5.2 Metoden HentSvarFraVSLightOgGemSomXmlFil

```
/// <summary>
```

```
/// På baggrund af et UUID der tidligere er modtaget forsøges svar beskeden hentet fra NemRefusion og gemt i en XML fil.
/// </summary>
/// <param name="uuid">Et gyldigt UUID</param>
/// <param name="stiTilXmlFil">Komplet sti til den fil som svaret ønskes gemt
i</param> public static void HentSvarFraVSLightOgGemSomXmlFil(string uuid,
string stiTilXmlFil) {
   // opret VSLight klient
   var vsLightKlient = new VSLightClient();
    try
   {
      // kald GetResponse
      var vsLightSvar = vsLightKlient.GetResponse(uuid);

      if (vsLightSvar == null) // intet svar endnu
      {
         Console.WriteLine("Intet svar fundet endnu, vi prøver igen senere. UUID: " + uuid);
         // Håndter.
      }
      else // svar modtaget, det gemmes i fil
      {
         // for at bevare gyldigheden af det signerede XML fra NemRefusion er det vigtigt med
DisableFormatting!!!          vsLightSvar.Save(stiTilXmlFil, SaveOptions.DisableFormatting);
Console.WriteLine("Svar fra VSLight skrevet til fil: " + stiTilXmlFil);
      }    }
   catch (MessageSecurityException mse) // sikkerhedsfejl
   {
      Console.WriteLine("Der er noget galt med klientcertifikatet: " + mse.Message);
   }
   catch (FaultException fe) // soap fejl med koder
   {
      switch (fe.Code.Name) // fejl skal håndteres ifølge fejlkoden
      {          case
"101":
            Console.WriteLine("Det angivne UUID er ugyldigt, det må vi lige gøre noget ved: " + fe.Message);
            // Håndter!
            break;
         case "102":
            Console.WriteLine("Det benyttede certifikat er ikke registreret hos VSLight: " + fe.Message);
            // Håndter!
            break;
         case "104":
            Console.WriteLine("VSLight kunne ikke få kontakt til NemRefusion, vi prøver igen senere: " + fe.Message);
            // Håndter!
            break;
         case "InternalServiceFault":
            Console.WriteLine("VSLight må være nede, vi prøver igen senere: " + fe.Message);
            // Håndter!
            break;
         default:
            Console.WriteLine("Uventet svarkode:" + fe.Message);
break;
      }    }
   catch (Exception e)
   {
      Console.WriteLine("Uventet fejl:" + e.Message);
   }
   finally // VSLight klienten skal altid lukkes
   {
      vsLightKlient.Close();
   } }
```